Software components (ROS, Python, and Linux)

I am going to direct you once again to the Appendix to see all of the software that runs the robot, but I'll cover the basics here to remind you. The base operating system for the robot is Linux running on a Raspberry Pi 3 SBC, as we said. We are using the ROS to connect all of our various software components together, and it also does a wonderful job oftaking care of all of the finicky networking tasks, such as setting up sockets and establishingconnections. It also comes with a great library of already prepared functions that we can justtake advantage of, such as a joystick interface.

The ROS is not a true operating system that controls the whole computer like Linux or Windows does, but rather it is a backbone of communications and interface standards and utilities that make putting together a robot a lot simpler. The ROS uses a publish/subscribe technique to movedata from one place to another that truly decouples the programs that produce data (such as sensorsand cameras) from those programs that use the data, such as controls and displays. We'll be makinga lot of our own stuff and only using a few ROS functions. Packt Publishing has several great booksfor learning ROS. My favorite is Learning ROS for Robotics by Aaron Martinez and Enrique Fernandez.

The programming language we will use throughout this book, with a couple of minor exceptions, will be Python. Python is a great language for this purpose for two great reasons: it is widely used in the robotics community in conjunction with ROS, and it is also widely accepted in the machine learning and AI community. This double whammy makes using Python irresistible. Python is an interpreted language, which has three amazing advantages for us:

- Portability: Python is very portable between Windows, Mac, and Linux. Usually theonly time you have to worry about porting is if you use a function out of the operatingsystem, such as opening a file that has a directory name.
- As an interpreted language, Python does not require a compile step. Some of the programs we are developing in this book are pretty involved, and if we write them in C or C++, would take 10 or 20 minutes of build time each time we made a change.You can do a lot with that much time, which you can spend getting your program torun and not waiting for make to finish.

- Isolation. This is a benefit that does not get talked about much, but having had a lotof experience with crashing operating systems with robots, I can tell you that the factthat Python's interpreter is isolated from the core operating system means that havingone of your Python ROS programs crash the computer is very rare. A computer crash means rebooting the computer and also probably losing all of your data you need todiagnose the crash. I had a professional robot project that we moved from Python toC++, and immediately the operating system crashes began, which shot the reliabilityof our robot. If a Python program crashes, another program can monitor that and restart it. If the operating system is gone, there is not much you can do without someextra hardware that can push the reset button for you. (For further information, referto Python Success Stories https://www.python.org/about/success/devil/).